commodore

COMMODORE BUSINESS MACHINES, INC.
901 CALIFORNIA AVENUE
PALO ALTO, CALIFORNIA 94304
TELEPHONE: (415) 326-4000 TELEX: 345-569
CABLE ADDRESS COMBUSMAC PLA

USR

The USR function allows a programmer to create a machine language subroutine which is callable from BASIC. USR has a parameter which is evaluated and placed in the floating accumulator at location $BØ. The format is as follows:

```
$BØ  -  exponent  +  $8Ø
                               normalized so
$B1  -  mantissa     MSB       B7 set

$B2  -     "

$B3  -     "

$B4  -     "           LSB

$B5  -  sign of mantissa

        Ø if mantissa = Ø

        + if mantissa  non-zero or plus

        - if mantissa  negative
```

The floating accumulator may be converted to a two byte integer in $B3 and $B4 (MSB, LSB) by a JSR $DØA7. On return to BASIC, an integer may be converted and passed in the floating accumulator. The MSB is loaded into the MOS 6502 accumulator A and the LSB into index register Y and then JSR $D278. Since the return address to BASIC is already on the stack and the integer-floating conversion might be the last step to execute, it is possible to do a JMP $D278 instead of a JSR $D278 and RTS.

Before executing USR from BASIC, locations 1 and 2 must be poked with the address, lo-hi, of the machine code subroutine. The address may be changed if the programmer desires to have more than one routine resident at one time.

It is recommended that the machine language subroutines be located in protected areas of RAM such as the unused tape buffer.

example: floating point representation

$1.5_{1Ø}$

8Ø   CØ ØØ ØØ ØØ ØØ

$BØ →

USR function example #1

```
ØØØØ   4C   3A   Ø3          JMP   USR
                             INT = $B3
                              *  = $33A
Ø33A   2Ø   A7   DØ    USR   JSR  FLPINT
033D   A5   B3               LDA  INT
Ø33F   A6   B4               LDX  INT+1             Swap bytes
Ø341   85   B4               STA  INT+1               to use
Ø343   86   B3               STX  INT                 as address
Ø345   A2   ØØ               LDX  #Ø                  indirect
Ø347   A1   B3               LDA  (INT,X)             load
Ø349   A8                    TAY                      LSB in Y
Ø34A   8A                    TXA                      MSB in A
Ø34B   4C   78   D2          JMP  INTFLP
                             INTFLP = $D278
                             FLPINT = $DØA7
```

USR function example:

X = USR(I)
-32768 ≤ I ≤ 32767
Ø ≤ X ≤ 255

Returns the contents of the byte whose address is specified by I. The variable I is preserved. Parameter is passed in the floating accumulator and translation is performed by appropriate BASIC subroutines.

```
10000 DATA 32,167,2Ø8,165,179,166,18Ø,133
10100 DATA 18Ø,134,179,162,Ø,161,179,168
10200 DATA 138,76,12Ø,21Ø
10300 FOR I = 826 TO 845
10400 READ N:POKE I,N
10500 NEXT
10600 POKE 1,58
10700 POKE 2,3
```

This is a BASIC program to POKE the USR machine language subroutine from the previous example into the memory. The hex codes have been translated into decimal and placed in data statements. The memory region used is the 2nd cassette data buffer area. Note locations 1 and 2 are poked with the start address of the subroutine:

3*256+3*16+1Ø = 826

USR function example #2

```
Ø33A    2Ø    A7    DØ    LOGB2   JSR    $DØA7      floating to integer
Ø33D    AØ    ØØ                  LDY    #Ø         LSB of result in
Ø33F    A5    B4                  LDA    $B4        LSB of integer
Ø341    6A                SHIFT   ROR    A
Ø342    9Ø    Ø5                  BCC    DONE       switch closed
Ø344    C8                        INY
Ø345    CØ    Ø8                  CPY     #8        no switches ?
Ø347    DØ    F8                  BVE     SHIFT
Ø348    A9    ØØ          DONE    LDA     #Ø        MSB in A = Ø
Ø34A    4C    78    D2            JMP     $D278     integer to floating
                                  *=Ø
ØØØØ    4C    3A    Ø3            JMP     LOGB2     vector for USR
```

```
10 PRINT USR(PEEK(59471)):GOTO 1Ø
```

Switches connected to USR port can be wired to cause a low logic level. The port can be PEEK'ed and this routine returns the bit #(0-7) or 8 if no switch is closed.